Mihályi Zoltán (MIZRAAI), Pinczel Máté (PIMRAAI)

Nagy kifejezőerejű lekérdező nyelvek rendezetlen (adat)fákhoz

Eredeti cikk:

Highly Expressive Query Languages for Unordered Data Trees

- Serge Abiteboul (INRIA & ENS Cachan, Serge.Abiteboul@inria.fr)
- Pierre Bourhis (University of Paris-Sud & Oxford University, bourhis@comlab.ox.ac.uk)
- Victor Vianu (U.C. San Diego & INRIA, vianu@cs.ucsd.edu)

Absztrakt

A cikk a rendezetlen adatfákat lekérdező nyelvek kifejezőerejét vizsgálja meg az AXML (Active XML) és a while nyelvcsalád kiterjesztésével. Ezek rendre mintaillesztésen alapulnak, bizonyos vezérlés hozzáadásával. A különböző variánsok kifejező ereje több tényezőtől függ, úgymint, az alapvető lekérdezések kifejezőereje, a számítások beékelésének módja az adatok közé, determinisztikus, vagy nemdeterminisztikus vezérlés. Egyes jellemzők komoly korlátozást jelenthetnek, de előnyük az erőteljes normál forma.

Bevezetés

Az utóbbi időkben nagy érdeklődés volt az adatfákon működő lekérdezőnyelvek iránt – köszönthetően az XML népszerűségének is. Ám ezek a kutatások nem a nyelv kifejező-erejére fektették a hangsúlyt, hanem a kiértékelés/végrehajtás hatékonyságára, így ezek statikus vizsgálata is egyszerűbb. Ez a cikk a más szempontból fontolja meg a problémát, a nyelvek kifejezőerejét szeretné felderíteni. Ráadásul rendezetlen adatfákon működő lekérdezőnyelveket vizsgál, amit a klasszikus adatbázisok halmaz-alapú megközelítése motivál. A vizsgált lekérdezőnyelvek mintaillesztést használnak, mint alapvető építőkövet, és ezekre építenek bizonyos vezérlést, így létrehozva egy komplex programot. Két fontos jellemzőt érdemes kiemelni, amelyek befolyásolják a kifejezőerőt: az alap lekérdezések (mintaillesztés) kifejezőerejét, és a „kódok" beágyazásának módját az adatfákba. Az utóbbihoz az AXML nyújtotta lehetőségeket használja a cikk, amit tiszta, rugalmas modellt nyújt az XML-be ágyazott kódok kezeléséhez. Vizsgálja továbbá a while nyelvcsalád kiterjesztését adatfákra. Ebben az esetben is jelentősen befolyásolják a kifejezőerőt a megengedett nyelvi konstrukciók, és a nyelv jellemzői. Betekintést enged az adatfák különlegességeibe, és néhány érdekes általánosítását is láthatjuk a klasszikus eredményeknek. Érdemes kiemelni az FO^k definiálhatóság fogalmát (azaz egy lekérdezés megfogalmazható-e elsőrendű logikában), amely az adatfáknál is hasznos eszköznek bizonyul a nyelvek kifejezőerejének megértéséhez.

A cikk fő eszköze, az AXML [1], az XML egy kiterjesztése beágyazott szolgáltatáshívásokkal. Ez sok esetben hasznosnak bizonyult már. Alapvetően egy magas szintű specifikációs keretrendszernek tervezték, adat-centrikus munkafolyamok leírásához. Jól illeszkedik a még alakulóban lévő dokumentumokhoz. Ebben a környezetben a kezdő és a végső fázis közötti kapcsolat érdekes. Ehhez kötődik a „dominance" [2] fogalma, amely egy egyszerű eszköz a kifejezőerő vizsgálatához, továbbá hasznos, amikor absztraháljuk a munkafolya-

matot, olyan módon, hogy egy részfolyamatot helyettesítünk egy szignatúrával, ami leírja a kapcsolatot a bemenet és a kimenet között.

Egy AXML példány – ahogyan a cikk használja – egy erdő, ami rang nélküli, rendezetlen fákból áll. A fák belső csúcsai véges ábécé feletti címkékkel vannak ellátva; a levelek pedig címkékkel, vagy adatokkal egy véges halmazból, vagy függvény szimbólumokkal. A függvények hívása, ahogyan a visszatérésük is őrfeltételekkel vannak levédve. Az erdők két akció folyamán alakulnak, ezek a függvényhívás és függvény visszatérés. Egy függvényhívás egy tiszta munkaterületet hoz létre a jelenlegi példányból egy lekérdezéssel. A munkaterület is tartalmazhat további függvényhívásokat, így megvalósítva a rekurzív hívást. A hívás eredménye egy erdő, ami a munkaterület végső állapotából egy lekérdezéssel jön létre. Az AXML tipikusan nemdeterminisztikus szemantikát alkalmaz, így az átmeneteket egy olyan tetszőleges függvény hívása, vagy visszatérése okozza, aminek az őrfeltétele igaz. Természetes módon definiálható a determinisztikus szemantika is, ami alatt azt értjük, hogy az összes hívás és visszatérés szimultán fut le, amelyeknek az őrfeltétele igaz. (Hasonlóan a Dataloghoz.) Tekinthetünk az AXML-re úgy, mint egy lekérdezőnyelvre, amelynek a bemenete egy kezdeti példány, és a kimenete egy fa az *Out* címkéjű csúcs alatt.

Az eredmények első csoportja arra az esetre koncentrál, amikor a függvények el vannak különítve az adatoktól, azaz csak a triviális beágyazást engedjük meg. A függvények csak adatokat manipulálnak, nem pedig részfákat. Kiderül, hogy ez ekvivalens a relációs nyelvekkel, amelyek Turing teljes programozási nyelvek, és beágyazott FO (első rendű) lekérdezéseket használnak. Ezek a nyelvek relációs automatákkal [3] vannak definiálva, vagy a while nyelvcsalád egész típussal kiegészített változatai. A Turing teljesség ellenére, ezek közel sem lekérdezés-teljesek, hiszen olyan egyszerű lekérdezéseket sem tudnak megválaszolni, mint a tartomány paritása, a 0-1 törvénynek [4] köszönhetően. A fákhoz definiált lekérdezőnyelv, a QAXML izolált függvények használatával kiterjesztett változata ekvivalens a while nyelv egész típussal kiterjesztett, adatfákon dolgozó változatával. Ezzel bizonyítható a kifejezőerejének korlátoltsága, de egy erős normálformát is ad a kiértékeléshez. Ez sokat segít az optimalizálásnál, mivel a bemenetet jelentősen redukálja, azon végzi el a számítást, és a végén alakítja vissza a valós adatokat. Ráadásul a logikai lekérdezések még rövidebben kiértékelhetőek, hiszen nem kell az utolsó lépés. (Hasonló módon, mint a relációs esetben.) Fontos eredmény az is, hogy ebben az esetben a nemdeterminisztikus megközelítés sem ad több lehetőséget.

Az izolált QAXML korlátoltsága a sűrű függvények bevezetésével enyhíthető, amely megengedi, hogy tetszőlegesen ágyazzunk be függvényeket. Ezzel a nemdeterminisztikus esetben bármilyen kiszámítható lekérdezést meg lehet fogalmazni, azaz lekérdezés-teljes. Ez azért van, mert ezzel ú.n. adat-nemdeterminizmust engedünk meg, azaz nemdeterminisztikusan választhatunk egy adatot egy halmazból.

Érdekes, hogy a determinisztikus esetet megvizsgálva kiderül, hogy ez nem lekérdezés-teljes a sűrű függvények használatával, tehát a nemdeterminizmus emelte a kifejezőerőt. Itt megjelenik a jól ismert „copy elimination problem”, eleve kizárva a teljességet.

Az eddigiekben a lekérdezésekben a változók atomi adatokat jelöltek, de a cikk megvizsgálja részfák megragadását is változókban. Ez megnöveli a kifejezőerőt, és még a determinisztikus, izolált QAXML is lekérdezés-teljessé válik így. Érdekes módon a nemdeterminisztikus variánsra ez nem igaz, az csak a gyengén nemdeterminisztikus lekérdezéseket képes kifejezni, amit az indokol, hogy a vezérlésben jelenik meg a nemdeterminizmus, nem pedig az adatok kiválasztásában. Hogy teljessé tegyük a nyelvet, több kell az izolált függvé-

nyeknél, habár a sűrű függvényekig nem kell elmenni. Elég a köztes állapotokban megengedni, a függvények nem triviális beágyazását.

Most is találhatunk kapcsolatot a while részfákat megragadó változata és a QAXML között. Ez a nyelv egyszerűbb, mivel nincs szükség az egész típusra és egyéb szerkezetekre. Továbbá a nemdeterminisztikus esethez nyerünk egy normálformát, ami minden nemdeterminizmust az utolsó lépésre hagy.

Kapcsolódó munkák

Az AXML vizsgálata több különböző technikával történik a klasszikus lekérdezőnyelvek elméletéből merítve, úgymint, elsőrendű lekérdezések korlátos számú változóval, normálformák, 0-1 törvények, nagy kifejezőerejű nyelvek.

A fákat lekérdező és transzformáló nyelveket széles körben megvizsgálták már az XML-hez kötődően, főleg az XQuery, XPath és XSLT egyes részeinek általánosításával [5,6,7]. Ezek közül sok adatok nélküli fákon dolgozik, de a legújabbak adatokkal (véges ábécé felett). Sok koncentrál a statikus elemzésre, azaz a kifejezőerőt feláldozza a könnyű kiszámítás miatt [8,9,10]. Egy kevés tanulmány található a nagy kifejezőerejű nyelvekről adatfákon, de ezek általában rendezett fákon működnek [11,12,13,14]. Ez a cikk rendezetlen fákkal dolgozik. Ez egyezik a relációs modell megközelítésével, ahol a rekordok sorrendje lényegtelen. E mögött az húzódik, hogy az információ számít, nem a reprezentáció, mint például a sorrendjük az elemeknek. A rendezés hiány lehetőségeket ad az optimalizációra és a párhuzamos feldolgozásra, és könnyíti a statikus elemzést. Ezek a különbségek teszik összehasonlíthatatlanná e cikket az említett munkákkal.

Alapfogalmak

Legyen **dom** adatok egy végtelen halmaza, és egy tőle diszjunk végtelen halmaz a változók halmaza. A relációs séma σ pedig relációk egy véges halmaza a megfelelő aritásokkal. σ egy példánya minden szimbólumhoz tartalmaz egy véges reláció a megfelelő aritással **dom** felett.

Az *FO lekérdezések σ* felett a következőképpen definiálhatóak: Egy atom $R(x_1,\ldots,x_m)$ és $x_1 = x_2$, ahol *R* egy reláció σ-ban m aritással és minden x_i vagy változó, vagy adat. A formulákat az atomok, logikai operátorok és kvantorok megszokott kombinálásával állíthatjuk elő. Az aktív tartomány szemantikát alkalmazzuk, ami a tartományokat lekorlátozza a példányban vagy a lekérdezésben előforduló értékekre.

Egy lekérdezőnyelv lekérdezés-teljes, ha minden kiszámítható lekérdezést le tud írni. Ez a klasszikus relációs környezetben feltételezi azt, hogy a válasz előállítása csak a bemenetből használ adatokat és determinisztikus.

Az FO lekérdezések nem lekérdezés-teljesek, például nem tudnak kifejezni olyan egyszerű lekérdezéseket, mint egy gráf tranzitív lezártja. Ez részlegesen megoldható rekurzió beépítésével. Ezek a kiterjesztett változatok a lekérdezések két robosztus osztályához tartanak, ezek a *fixpont* lekérdezések és a *while* nyelvcsalád változatai. A while az FO-t kiterjeszti (i) reláció változókkal, aminek értékül lehet adni a lekérdezések eredményeit (destruktív szemantikával) és (ii) egy ciklussal a „$\text{while } \mathrm{R} \neq \emptyset \text{ do}$” formában. A fixpont lekérdezések a while$^+$ segítségével írhatók le. Ez az értékadást kumulatív módon valósítja meg, és a ciklust a „while change do” formában valósítja meg. A kumulatív értékadás miatt a relációk egyre csak nőnek, és a ciklus akkor áll meg, ha az előző iterációban már nem került újabb rekord a relációhoz, azaz elértük a fixpontot. Az ezen a nyelven kifejezett lekérdezések mind kiszámíthatóak polinom időben a bemenet méretének függvényében, míg a while nyelven megfogal-

mazott lekérdezések PSPACE beli problémák. E korlát áttörésének egyik lehetősége, hogy Turing teljessé tesszük a nyelvet az egész számok típusának bevezetésével és egy „while i > 0 do" szerkezetű ciklus hozzáadásával. Ezt a nyelvet $while_{\mathbb{N}}$-nel jelöljük. Ez a nyelv teljes a rendezett adatbázisokon. De a tartomány paritását már nem tudja kiszámítani, ugyanis erre is igaz a 0-1 törvény: annak az esély, hogy formula igaz lesz egy n méretű bemenetre, 0-hoz vagy 1-hez konvergál, ha $n \to \infty$.

A $while_{\mathbb{N}}$ és társai kifejezőerejét egy normálforma világítja meg, ami polinom időben kiszámíthatóvá teszi a kiértékelését ezeknek a programoknak egész számokon. Természetesen az egész számok megfelelnek a rekordok ekvivalencia osztályainak, amiket a program szervez össze. Pontosabban képzeljünk el egy $while_{\mathbb{N}}$ programot, ami adatok egy véges C halmazára hivatkozik, és aminek az FO lekérdezései maximum k változót használnak. Könnyű észrevenni, hogy minden reláció, amit a program a végrehajtása során létrehoz, definiálható FO^k formulák összefűzésével. Vegyünk egy I példányt, konstansok egy C halmazát, és legyen $\equiv_{I,k,C}$egy ekvivalencia reláció a rekordokon, melyeknek aritása $l \leq k$ a következő képpen: Minden $\varphi \in FO^k$-ra, amely C adatait használja fel és l szabad változója van, $\bar{a} \in \varphi(I) \Leftrightarrow \bar{b} \in \varphi(I)$. Ekkor létezik olyan fixpont lekérdezés, amely kiszámítja a következőket:

- az ekvivalencia osztályai $\equiv_{I,k,C}$-nek
- egy rendezése az ekvivalencia osztályoknak.

A definíció következménye, hogy minden reláció, ami I-ből lett képezve egy FO^k formulával, a megfelelő ekvivalencia osztályok uniója. Minthogy az osztályok rendezettek, tekinthetjük őket egész számoknak is. Viszont így el kell tudnunk végezni a műveleteket az egészekkel, a nélkül, hogy visszakeresnénk az osztályokat. Ehhez megfelelő információt el kell eltárolnunk róluk. Egyszerűen belátható, hogy létezik egy olyan véges F^k halmaz, ami konjunktív formulákat tartalmaz, maximum k db változóval, hogy minden FO^k formula az adott séma felett kiértékelhető az F^k formuláinak és negáltjaiknak unióival. Legyen minden $q \in F^k$-ra a(q) az atomok száma q-ban. Bebizonyítható, hogy létezik egy fixpont lekérdezés, ami kiszámít minden $q \in F^k$-hoz egy Action_q relációt, ami minden a(q) hosszú rendezett sorozatához az $\equiv_{I,k,C}$ ekvivalencia osztályainak, megadja az eredményét annak, ha q-t alkalmazzuk erre a sorozatra. Így az $\text{Action}(\text{I}, \text{k}, \text{C}) = \{\text{Action}_q \mid \text{q} \in \text{F}^k\}$ biztosítja a megfelelő információkat az FO^k lekérdezések kiértékeléséhez az egész számokon.

Definíció: (Fa)

Az adatfák címkézettek, rang nélküliek, rendezetlenek. Legyenek a következő halmazok végtelenek és diszjunktak: a csúcsok N (n,m,…), a címkék Σ (a,b,c,…), adatok $D(\alpha, \beta, \dots)$. A fa egy véges bináris reláció N felett, ahol minden csúcsnak pontosan egy szülője van, kivéve a gyökérnek; és egy címkéző függvény, ami minden csúcshoz egy címkét vagy adatot rendel, de adatot csak levélhez. Feltesszük, hogy a fa redukált (azaz nincsen két címkék és adattagok tekintetében izomorf testvér részfa). Ez a halmaz megközelítésnek (nem a zsáknak) felel meg a relációs adatbázisokban. Az I fában előforduló adatok halmazát $dom(I)$-vel jelöljük.

Definíció: (C-genericitás)

Legyen C adatok egy véges halmaza. Egy $\mathcal{R}$ reláció (fák felett) a Σ címkékkel C-generikus, ha minden olyan ρ injektív leképzésre $N \cup D \cup \Sigma$ felett, melyre $\rho(N) \subseteq N, \rho(D) \subseteq D$ és ρ identitás $C \cup \Sigma$ felett, $(I, J) \in \mathcal{R} \Leftrightarrow (\rho(I), \rho(J)) \in \mathcal{R}$.

Definíció: (Kiszámíthatóság)

A kiszámíthatóság fogalma a szokásos: Egy $\mathcal{R}$ reláció kiszámítható, ha létezik nemdeterminisztikus Turing gép úgy, hogy adott egy $\leq$ rendezés az adatokon és egy $enc_{\leq}(I)$ kódolása a bemenő fának a gép szalagján, ekkor a gép pontosan akkor terminál a kimenetén $enc_{\leq}(J)$-vel, ha $(I, J) \in \mathcal{R}$.

Definíció: (DTD)

Egy megszorítás a bementi fák szerkezetére. Δ-val jelöljük.

Definíció: (Fa lekérdezés)

Egy fa lekérdezés egy kiszámítható, C-generikus $\mathcal{R}$ reláció, amely a Σ feletti Δ-nak megfelelő fákról képez a Σ feletti fákra úgy, hogy minden$(I, J) \in \mathcal{R}$: (i) $dom(J) \subseteq dom(I) \cup C$, és I és J csúcsainak halmaza diszjunkt.

Definíció: (Lekérdezés-teljes)

Ez lekérdezőnyelv lekérdezés-teljes, ha minden kiszámítható lekérdezést kifejez.

Akkor mondunk egy lekérdezést determinisztikusnak, ha egyértelmű választ ad minden bemenetre a csúcsok neveitől eltekintve.

Eredmények

AXML lekérdezőnyelvek

Legyen $\mathcal{F}$ a függvény nevek egy végtelen halmaza, $\mathcal{F}^! = \{!f \mid f \in \mathcal{F}\}$ és $\mathcal{F}^? = \{?f \mid f \in \mathcal{F}\}$ függvény szimbólumok halmazai! Természetes módon $!f$ jelöli a helyet, ahol egy függvényt meg lehet hívni, és $?f$, ahova egy függvény visszatér az eredménnyel. Miután egy csúcsban az eredményt megkaptuk, a csúcsot töröljük. Egy AXML fa olyan, hogy a belső csúcsai fel vannak címkézve Σ elemeivel, a levelei pedig Σ elemeivel, vagy függvény szimbólumokkal, vagy adatokkal. Az AXML fa is redukált, azaz nincs két olyan testvér, ahol nincs futó függvényhívás, de mégis izomorfak.

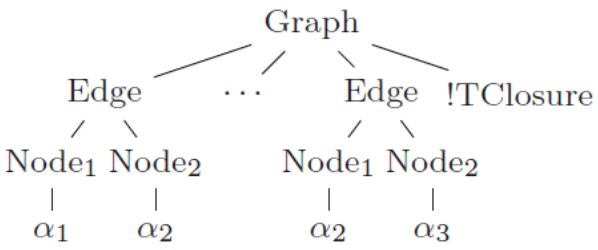

Figure 1: AXML tree

DTD (Document Type Declaration)

A bemenő fa szerkezetét írja le. Rendezetlen fákon csak annak van értelme, hogy előírjuk, milyen címkéjük lehet a gyermekeknek. Így a $|b| \geq k$ formájú kifejezések logikai kombinációit használjuk, ahol $k \geq 0$ egész, és $b \in \Sigma \cup \mathcal{F}^! \cup \mathcal{F}^? \cup \{dom\}$. Feltesszük, hogy az összes DTD kiteszi, hogy r legyen a fa csúcsa. Egy DTD statikus, ha nem enged meg függvény szimbólumokat, egyébként aktív.

Minták

A minták a lekérdezőnyelvek alapkövei.

Egy P minta egy $(T, cond)$ rendezett pár.
Kétféle változót használunk: a *strukturális változókat* (V,W,…), amik a címkével vagy függvény szimbólummal ellátott csúcsokra illeszkednek; és az *adat változók*, amik az adatokkal ellátott csúcsokra illeszkednek.
T egy fa-minta, azaz egy oylan fa, aminek a csúcsai egyedi változókkal vannak címkézve, és az élei pedig a *gyermek* (/) , vagy *leszármazott* (//) címkékkel vannak ellátva, ahol a leszármazott reláció reflexív.
A csúcsok lehetnek negatívak (¬), vagy pozitívak (alapértelmezetten pozitív). A gyökér mindig pozitív.
Egy csúcsot *határcsúcsnak* nevezünk, ha az a gyökér, vagy egy negatív csúcs.
T minden S részfájára, amelynek gyökere nem negatív, az S^+ fát a negatív csúcsok eltávolításával kapjuk.
Legyen var egy határcsúcsokból a változók halmazába képező függvény, a következő:
$var(r) = T^+$ csúcsainak halmaza, és tetszőleges b-re
$var(b) = \bigcup_{b' felmenője\ b-nek} var(b') \cup S_b^+$ $változói$, ahol S_b^+ a b gyökerű részfa pozitív gyökérrel.
A $cond$ feltétel minden határcsúcshoz egyenlőségek egy logikai kombinációját társítja a következő formában:

- $V = t$, ahol V strukturális változó és t címke, vagy függvény szimbólum
- $X = Y$, ahol X adatváltozó és Y adatváltozó, vagy adat.

Legyen P egy $(T, cond)$ minta. P illesztését egy I erdőhöz a következő módon definiáljuk strukturális rekurzióval. Legyen ν egy leképezés $var(T^+)$-ról I csúcsaira:

- A gyermek és a leszármazott relációt megtartja.
- Minden X adatváltozóhoz $\nu(X)$ egy csúcs egy adattal.
- $cond(r)$-t kielégíti, azaz $V = t$-ből következik, hogy $\nu(V)$ címkéje t, és $X = Y$-ból következik, hogy $\nu(X)$ és $\nu(Y)$ címkéje azonos (hasonlóan, ha Y adat).
- Minden maximális N részfájára T-nek, aminek a gyökere egy negatív b igaz, hogy ν-nek nincsen kiterjesztése $T \oplus N$-re, ahol $T \oplus N$-et T-ben b pozitívra állításával kapjuk, hogy ν kielégíti $cond(b)$-t.

Egy adott I AXML erdőre és egy P mintára $Bind(P, I)$-vel jelöljük ezen illesztések halmazát.
Azt mondjuk, hogy $I \vDash P$, ha $Bind(P, I) \neq \emptyset$.

Van, hogy a mintát egy kitüntetett csúcshoz képest akarjuk illeszteni. Ekkor egy $V = self$ formájú kifejezést használunk a $cond$-ban. Ez egy relatív kifejezés, amit egy (I, n) páron értékelünk ki, ahol I egy erdő, n pedig egy csúcsa. Ekkor nyílván $\nu(V) = n$.

Mintaillesztéses lekérdezés

A mintaillesztéses lekérdezés $Body \rightarrow Head$ formájú szabályok egy halmaza. Itt $Body$ egy minta, $Head$ pedig egy fa, aminek a belső csúcsai címkékkel, a levelei pedig vagy címkékkel, vagy $\mathcal{F}^!$ elemeivel, vagy adatváltozókkal vannak ellátva $Body^+$-ból. Továbbá minden $Head$-ben megtalálható változó a konstruktor csúcs alatt fordul elő (jel: {X}). A szabály eredményét a $Body$ I-re való illesztéseinek segítségével kapjuk meg, ehhez a $Head$-ben a konstruktor csúcsban található T részfát kell a helyettesíteni egy erdővel, ami minden $\nu \in Bind(Body, I)$-re tartalmazza T egy új másolatát, amiben az X adatváltozót a $\nu(X)$ adattal helyettesítjük. A válasz a mintaillesztéses lekérdezésre a szabályokra adott válaszok uniója.

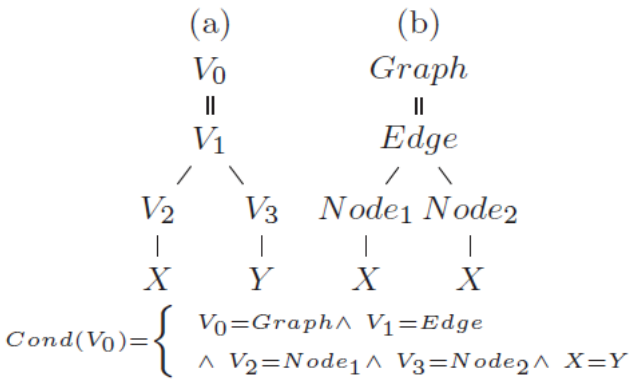

Figure 2: A simple pattern: full specification (a) and concise version (b)

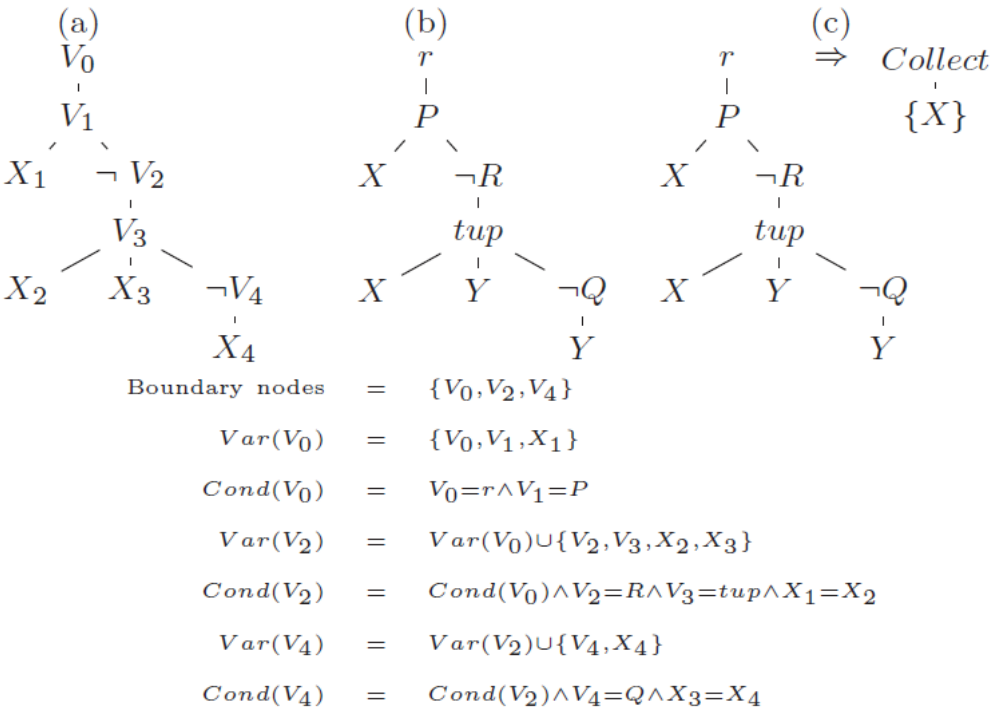

Figure 3: A complex pattern: (a) full specification (b) concise version (c) a query using the pattern

Programok és példányok

Egy QAXML program Q egy (Φ, Δ) páros, ahol Δ egy DTD, Φ pedig függvénydefiníciók egy halmaza. Minden $f \in \mathcal{F}$-hez $a_f$legyen egy egyedi címke. Ez lesz a függvény munkaterületének gyökere, ahol f kiértékelődik. Egy függvény specifikációja 4 részből áll:

- belépési őrfeltétel: Ha igaz, akkor hívható a függvény.
- bemenő lekérdezés: Egy relatív lekérdezés, ami inicializálja a munkaterületet. Ekkor a self-hez hozzá lesz rendelve a hívás helye.
- visszatérési őrfeltétel

- kimenő lekérdezés: Egy lekérdezés, aminek a gyökere a_f.

Egy AXML példány I egy (τ, eval) pár, ahol τ egy AXML erdő, és eval egy injektív függvény, ami τ ? f-fel címkézett csúcsairól képez a_f gyökerű részfáira τ-nak, és minden $a_f = eval(n)$ valamely n-re.

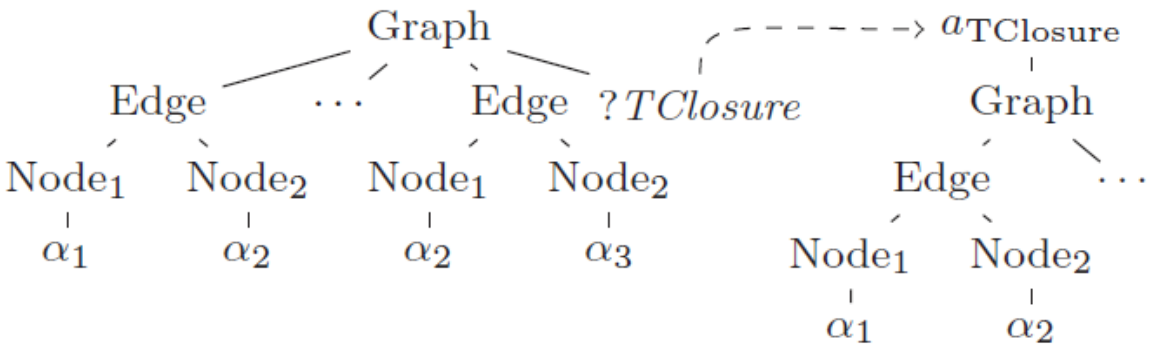

Figure 4: An AXML instance with an eval link

Nemdeterminisztikus szemantika

Legyenek $I = (\tau, \text{eval})$ és $I' = (\tau', \text{eval}')$ példányok. Az I' az I lehetséges következő példánya (jel: $I \vdash I'$), akkor és csak akkor, ha I' előállítható I-ből egy függvény meghívásával, vagy eredmény visszatérítésével, ahol az őrfeltétel igaz.

Mikor !f-et meghívjuk n csúcsban, akkor n címkéjét ?f-re változtatjuk és eval-hoz hozzáadunk egy (n,T') elemet, ahol T' az a_f gyökerű fa, amit a bemenő lekérdezés (τ, n)- en való kiértékelésével kapunk. Amikor a válasz az n csúcsra megérkezik, akkor a fák, mint n testvérei adódnak a fához, majd n törlődik. Ez csak akkor lehetséges, ha eval(n) nem tartalmaz futó függvényhívást.

A Q = (Φ, Δ) egy számítása egy maximális $\{(I_i)\}_{0 \le i < n}$, hogy $n \in \mathbb{N} \cup \{\omega\}, I_0$eleget tesz Δ-nak és minden i-re, $0 < i < n$, $I_{i-1} \vdash I_i$. A számítás akkor terminál, ha véges.

Determinisztikus szemantika

Minden olyan függvényt szimultán meghívunk/visszatérünk belőle, amelynek az őrfeltétele igaz.

Ha a lekérdezés terminált, akkor az *Out* címke alatt található az eredmény.

QAXML izolált függvényekkel

Definició:

Egy QAXML program izolált függvényekkel egy $Q = (\Phi, \Delta)$ pár, ahol Δ egy statikus DTD és minden Φ beli $Body \rightarrow Head$ szabályban a $Head$-ben a konstruktor csúcs alatt nincsen függvény szimbólum. Egy I példányra, ami kielégíti Δ-t, legyen $I^!$ az I egy plusz ! f csúccsal a gyökere alatt. A Q program egy $\mathcal{R}$ lekérdezést fejez ki Δ DTD-vel, ha minden Δ-nak eleget tevő I-re: $(I, O) \in \mathcal{R} \Leftrightarrow$ létezik olyan számítása Q-nak, $I^!$en hogy terminál ,és az Out címke alatt O található.

Izolált függvények és FO^k definiálhatóság

Vegyünk egy I példányt és egy Q programot, ami konstans adatok egy véges C halmazát használja a mintáiban. Képzeljük el a program futását, ahogyan a gyökér alá egy másik részfa generálódik, de ezen kívül az I változatlan marad. A minták egy része az I-re illesztődik, a többi az I-n kívüli munkaterületek fáira. Az I-re való leképezéseket előre ki lehet számolni a releváns részmintákra, és eltárolni egy relációs struktúrában, $\sigma(I)$-ben.

A szabályok fejrészéből generált részfákat ezekből a leképzésekből generáljuk. Hívjuk a konstruktor csúcsokból példányosított csúcsokat kiterjesztett csúcsoknak. Legyen $R_\varepsilon$az a reláció, ami azokból a leképzésekből áll, amelyeket egy lépésben a kiterjesztett csúcsok létrehozásához használtunk.

Megmutatható, hogy *létezik olyan $k > 0$ (csak Q-tól függően), hogy minden R_ε definiálható a $\sigma(I)$-ből egy FO^k formulával és a C halmaz elemeivel*.

Ennek a bizonyítása/magyarázata megtalálható a cikkben.

Ebből levezethető, hogy ez a nyelv ekvivalens a $while_{\mathbb{N}}$ fákra használt változatával.

A $while_{\mathbb{N}}$ nyelvek fákhoz

Először a nyelv nemdeterminisztikus változatával foglalkozunk.
A nyelv egész számokat: i, j, … (0-ra inicializálva); és erdő változókat: X, Y, … használ.
Van két kiemelt erdő változó, az *In* és az *Out*. Továbbá van egy verem, amibe erdő típusú változókat lehet helyezni.

Az alap utasítások a következők:

- egész szám növelése/csökkentése
- X := {T}, ahol X erdő, T pedig konstans fa, függvények nélkül.
- X := Q(Y), ahol X és Y erdő változók, és Q egy lekérdezés.
- X := Y ∪ Z, ahol X, Y ,Z erdő változók, mind különbözik In-től.
- X := a[Y], ahol Y, X In-től különböző erdők, és „a” egy címke; az utasítás a csúcs alá fűzi Y fáit, és ezt X-nek adja értékül.
- push(X), ahol X erdőváltozó különbözik In-től.
- X := top, kiszedi a verem felső elemét, és X-nek adja értékül.

A program vagy egy utasításból áll, vagy a következő konstrukciós eszközöket használja:

- while i > 0 do program
- while X ≠ ∅ do program
- program1; program2 (kompozíció)
- program1 | program2 (nemdeterminisztikus választás)

A programnak szintén van egy DTD megszorítása az inputokra.
A kimenet az *Out* változó végső értéke.
A program megvalósítja az $\mathcal{R}$ lekérdezést, ha minden DTD-nek megfelelő I bementre a lehetséges kimenetek halmaza éppen $\{J \mid (I,J) \in \mathcal{R}\}$.

A determinisztikus változat a nemdeterminisztikus választás elhagyásával kapható.

NQAXML izolált függvényekkel

Tétel:
A NQAXML izolált függvényekkel pontosan azt fejezi ki, mint amit az N-$while_{\mathbb{N}}^{tree}$.

Az N-$while_{\mathbb{N}}^{tree}$ program szimulációja NQAXML-lel egyszerű.
A NQAXML program szimulációja a következő lépésekből áll:

- A relációs struktúra $\sigma(I)$ kiszámítása I-ből.
- Az adott k-ra $\sigma(I)$-ből a rendezett ekvivalenciaosztályok, és az Action(I,k,C) kiszámítása, ahol C a Q-ban említett konstansok halmaza.
- $\sigma(I)$ és Action(I,k,C) reprezentációjának kiszámítása Turing gép szalagjára, ahol az ekvivalenciaosztályokat egészek jelölik.
- a Turing gép szimulálása, ami kiszámolja a választ a szalagra.
- Minden terminált számításhoz a kimenet kiszámítása a szalagról.

Tétel:
Minden NQAXML programhoz izolált függvényekkel, létezik egy ekvivalens Q_{nf} program, ami hatékonyan előállítható Q-ból, és a következő számításokat végzi I-n:

1. Polinom időben előállítja az $\sigma(I)$ relációs szerkezet, az ekvivalenciosztályok és az Action(I,k,C) fa reprezentációját.
2. Elvégez egy tetszőleges számítást a példány reprezentációján.
3. Polinom időben előállítja az eredményt.

A NQAXML nem lekérdezés-teljes. Legyen a DTD a következő: $r \to a\ a, a \to |dom| \geq 0$.
Ekkor nem tudjuk kiíratni az egyik halmazt nemdeterminisztikusan választva.

DQAXML izolált függvényekkel

Mivel a $while_{\mathbb{N}}^{tree}$-t az N-$while_{\mathbb{N}}^{tree}$-ből kapjuk a nemdeterminisztikus választás elhagyásával, ezért nyílván annak egy részhalmazát fejezi ki.

Tétel:
A $while_{\mathbb{N}}^{tree}$ pontosan a determinisztikus részét fejezi ki annak, amit az N-$while_{\mathbb{N}}^{tree}$ kifejez.

Tétel:
A $while_{\mathbb{N}}^{tree}$ pontosan azt fejezi ki, amit DQAXML izolált függvényekkel kifejez.

Tétel:
A DQAXML izolált függvényekkel pontosan a determinisztikus részét fejezi ki annak, amit az NQAXML izolált függvényekkel kifejez.

Logikai értékű lekérdezések

Gondoljunk egy NQAXML programra. Azt mondjuk, hogy logikai értékű, ha egy olyan fát ad vissza, ami egy csúccsal rendelkezik amihez *accepted*, vagy *rejected* címke tartozik. Egy I bemenetet Q elfogad, ha legalább egy számítással elfogadja azt. A logikai N-$while_{\mathbb{N}}^{tree}$, QAXML és $while_{\mathbb{N}}^{tree}$ programok analóg módon vannak definiálva.
Azt mondjuk, hogy két logikai értékű program ekvivalens, ha ugyanaz a DTD-jük, és ugyanazokat a példányokat fogadják el.

Tétel:
Az alábbi nyelvek ugyanolyan kifejezőerővel bírnak a logikai értékű programok tekintetében:

- NQAXML és DQAXML izolált függvényekkel
- N-$while_{\mathbb{N}}^{tree}$ és $while_{\mathbb{N}}^{tree}$ akár a verem, az X := Y ∪Z és az X := a[Y] utasítások nélkül is

Természetesen a normálformánk is erősebb, hiszen az utolsó lépés (az adatok visszaalakítása) elhagyható.

QAXML sűrű függvényekkel

Definició:
Egy QAXML program sűrű függvényekkel egy $Q = (\Phi, \Delta)$ pár, ahol Φ függvény definíciók egy halmaza, és Δ egy statikus DTD. Egy I példányra, ami kielégíti Δ-t, legyen az $I^{!*}$ az I kiterjesztése minden címkével (nem adat és nem függvény) ellátott csúcs alatt egy $!f$ függvényhívással. A Q program egy $\mathcal{R}$ lekérdezést fejez ki Δ DTD-vel, ha minden Δ-nak eleget tevő I-re $(I, O) \in \mathcal{R} \Leftrightarrow$ létezik olyan számítása Q-nak, $I^{!*}$en hogy t és az *Out* címkel alatt pontosan O van.

Tétel:
Az NQAXML sűrű függvényekkel lekérdezés-teljes.

Tétel:
Az DQAXML sűrű függvényekkel nem lekérdezés-teljes.

QAXML fa változókkal

Ez a kiterjesztés a fa részváltozóinak megragadásán alapul. Ez az alap lekérdezések kifejezőerejét növeli meg olyan mértékben, hogy már a determinisztikus esetben, izolált függvények használatával is lekérdezés-teljessé válik a nyelv.

Tétel:
Az $DQAXML^{\tau}$ izolált függvényekkel lekérdezés-teljes.

Ellenben a nemdeterminisztikus esetben nem teljes, hiszen csak az ú.n. *gyengén nemdeterminisztikus* lekérdezéseket fejezi ki. De nem kell elmennünk a sűrű függvények használatáig, elég azt kikötni, hogy a kezdetben csak a gyökér alatt szerepelhet függvényhívás, de később a konstruktor csúcsok alatt is megengedett. Ezt hívjuk query-dense függvényhívásnak.

Tétel:
Az $NQAXML^{\tau}$ query-dense függvényekkel lekérdezés-teljes.

While fa változókkal

Definiáljuk a $while^{\tau}$-t a következő jellemzőkkel:

- X,Y,Z, … rendre erdő változók
- X := $\varphi(Y)$, ahol X változó, Y változó vagy konstans fa, φ egy minta
- while X $\neq \emptyset$ do
- (nemdet. esetben: program1 | program2 egy nemdet. választás)
- *In* és *Out* változók a szokott módon.

Tétel:
A $while^{\tau}$ ekvivalens a $DQAXML^{\tau}$ izolált függvényes változatával és a N-$while^{\tau}$ ekvivalens a $NQAXML^{\tau}$ izolált függvényes változatával.

A nem determinisztikus eset kiterjeszthető egy X := choose(Y) művelettel, mely nemdeterminisztikus módon választ egy fát Y erdőből, és azt adja X-nek értékül. Ez az N^{d}-$while^{\tau}$.

Tétel:
Az N^{d}-$while^{\tau}$ lekérdezés-teljes és ekvivalens az $NQAXML^{\tau}$ query-dense változatával.

Ráadásul elég a *choose* műveletet csak a lekérdezés végén használni. (normálforma)

További kutatási terv

A jövőben megvizsgálandó/érdekes problémák, amik a cikk folyamán felmerültek:

- meghatározni az izolációs feltételeket, amik mellett az eredmények, még mindig igazak maradnak
- részletesebben foglalkozni a DTD hatásával a teljesség szempontjából
- meghatározni a DTD-ket, amikkel a QAXML izolált függvényekkel megtartja a 0-1 törvényt
- találni egy természetes determinisztikus lekérdezés-teljes nyelvet találása

A szerzők a jövőben meg kívánják vizsgálni a automaták átalakítók használatát. (Ez hasonló lehet az egész számokra való leképzéshez a bevezetésben.)

+ Érdemes lehet elgondolkodni a módszer hatékony megvalósításán, a megfelelő nyelv kiválasztásán.

Irodalomjegyzék

(Nem egyezik meg a cikk forrásaival, kiegészítő információkat is ad. A kivonat szerkezete a cikkéhez hasonló, így a cikk forrásainak az eredetiben utána lehet nézni.)

[1] http://webdam.inria.fr/axml/index.axml.html#documents

[2] D. Calvanese, G. D. Giacomo, R. Hull, and J. Su. Artifact-centric workflow dominance. In ICSOC/ServiceWave, 2009.

[3] http://sanskrit.inria.fr/huet/PUBLIC/Pune_tutorial.pdf

[4] http://en.wikipedia.org/wiki/Kolmogorov's_zero-one_law

[5] W. Janssen, A. Korlyukov, and J. V. den Bussche. On the tree-transformation power of xslt. Acta Inf., 43(6), 2007.

[6] F. Neven. Automata, logic, and XML. In CSL, 2002.

[7] T. Schwentick. Automata for XML - a survey. J. Comput. Syst. Sci., 73(3), 2007.

[8] M. Bojanczyk. Automata for data words and data trees. In RTA, pages 1–4, 2010.

[9] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In CSL, pages 41–57, 2006.

[10] L. Segoufin. Static analysis of xml processing with data values. SIGMOD Record, 36(1):31–38, 2007.

[11] M. Benedikt and C. Koch. From XQuery to relational logics. ACM Trans. Database Syst., 34(4), 2009.

[12] J. Hidders, S. Marrara, J. Paredaens, and R. Vercammen. On the expressive power of XQuery fragments. In DBPL, 2005.

[13] J. Hidders, J. Paredaens, R. Vercammen, and S. Demeyer. A light but formal introduction to XQuery. In XSym, 2004.

[14] C. Koch. On the complexity of nonrecursive XQuery and functional query languages on complex values. ACM Trans. Database Syst., 31(4), 2006.